Amendments to Claims

This listing of claims will replace all prior versions and listings of claims in the application:

Listing of Claims

1. (currently amended) A method for developing a dataflow application comprising:

developing one or more data transformations using a host language;

assembling several data transformations having ports into a map component with links between ports using a declarative language for static assemblage and a host language for dynamic assemblage, wherein a map component performs a respective data transformation;

compiling one or more map components with syntactic and semantic analysis; and

synthesizing the compiled map components into an executable dataflow application including removing the design time links between ports.

2. (original) The method of claim 1, wherein the steps of creating an executable dataflow application comprises:

synthesizing a hierarchical map component;

flattening the hierarchical map component; and

determining the executable dataflow application using the flattened hierarchical map component in conjunction with runtime properties.

3. (original) The method of claim 1, further comprising encrypting the dataflow graphs prior to the step of compiling the map component.

4. (currently amended) The method of claim 1, wherein the host language for data transformation logic is ~~Java~~ an object-oriented programming language.

5. (currently amended) The method of claim 1, some of the map components implementing dynamic assemblage logic implemented in ~~Java~~ an object-oriented programming language.

6. (original) The method of claim 1, some of the map components comprising a plurality of other map components arranged hierarchically.

7. (original) The method of claim 1, some of the map components being static which consistently generate the same hierarchical map.

8. (original) The method of claim 1, some of the map components being dynamic to generate different hierarchical maps dependent on properties and dynamic logic.

9. (original) The method of claim 8, the dynamic map components receiving information concerning properties, port types, and dataflow graph implementation and configuring its properties and internal dataflow graph implementation based on said received information.

10. (original) The method of claim 1, one or more of the map components having interface and implementation properties.

11. (original) The method of claim 1, some of the map components ports configured for sending data and some configured for receiving data.

12. (original) The method of claim 11, some of the ports being typed based on the type of data conveyed.

13. (original) The method of claim 11, some of the ports being composite, comprising a plurality of hierarchical ports.

14. (currently amended) A computer-readable medium having a dataflow development system comprising:

a library of components, some of the components being scalar and comprising a data transformation, some of the components being composite, and comprising a hierarchy of other components representing data transformations;

a compiler to develop and assemble components into ~~a~~ an executable dataflow application linking components subject to schema and properties constraints;

an executor which executes the dataflow application in parallel; and

tools for accessing the functionality of the compiler, executor and component library.

15. The system of claim 14, some of the components being encrypted and comprising a proprietary data transformation.

16. The system of claim 14, some of the components including ports for accepting and producing data whereby the map components are linked.

17. (currently amended) A computer-readable medium having a dataflow application development environment where map components are selected from a plurality of reusable map components each representing one or more data transformations, the selected map components are visually assembled into a dataflow application, patterns of parallelism recognized, and at least some of the components executed in parallel by assigning a number of threads to a respective number of data transformations, and at least some of the components are scalar components with each scalar component executed on a separate thread.

18. The environment of claim 17, wherein the map components define properties affecting design behavior.

19. The environment of claim 17, wherein the map components define properties affecting execution behavior.

20. The environment of claim 17, where the map components are assembled using ports for conveying data and declared as link points to other map components.